

APPLICATION
FOR
UNITED STATES LETTERS PATENT
ENTITLED

SMART-CACHING SYSTEM AND METHOD

TO WHOM IT MAY CONCERN:

BE IT KNOWN THAT Alexandre K. Shah of 4222 Cordobes Cove,
San Diego, 92130, invented certain new and useful improvements
entitled as set forth above of which the following is a
specification:

PATENT GROUP
FOLEY, HOAG & ELIOT LLP
ONE POST OFFICE SQUARE
BOSTON, MA 02109-2170
TEL: 617-832-1000
FAX: 617-832-7000

2
3 SMART-CACHING SYSTEM AND METHOD

4
5 BACKGROUND OF THE INVENTION

6 (1) Field of the Invention

7 The present invention relates generally to expedited data
8 retrieval and processing, and more particularly to efficient
9 caching techniques.

10 (2) Description of the Prior Art

11 A cache is commonly known as a memory or another type of
12 storage device, utilized to store data on a temporary basis. The
13 degree of the term "temporary" can vary widely depending upon the
14 application and configuration of a system utilizing the cache.
15 For example, for an internet user utilizing a personal computer
16 (PC) to browse the internet, web pages requested by the user are
17 retrieved from an original server on the internet, and thereafter
18 often stored in the user's internet browser's cache directory on
19 the user PC's hard disk. By utilizing a cache to store requested
20 internet web pages, subsequent requests for the same web page can
21 be processed by retrieving the web page from the cache rather
22 than the original server, hence relieving the internet of
23 additional traffic and providing a more rapid response to the
24 user.

1 The term "cache memory" often refers to random access memory
2 (RAM) that a computer microprocessor can access more quickly than
3 typical RAM. Alternately, "disk cache" is intended to improve
4 the speed for writing and reading to/from a storage device such
5 as a hard drive, and disk cache can be part of the hard disk, or
6 a specified portion of RAM. In some systems where internet
7 accessibility, connectivity, and content requests are prevalent,
8 caching can be implemented across multiple servers wherein the
9 server caches are periodically refreshed.

10 Caches can be implemented at the user level to serve a
11 specific user, or caches can be designed to serve larger
12 combinations of user. For example, macro caches can be
13 designated on systems serving international, national, regional,
14 or organizational users to preserve, periodically update, and
15 make available for distribution, information anticipated to be
16 highly requested by users of the system. Similarly, local server
17 caches can be designed for corporate LANs, access provider
18 servers, etc., for example, to similarly cache information
19 determined to be more popular at a more localized level.

20 As the popularity of the internet increases, the need for
21 more efficient caching is desired to decrease processing times
22 and bandwidth demands on the internet.

1 There is currently not an efficient method of caching
2 information that can be easily adapted for the increasing demands
3 provided by systems or networks such as the internet.

4 What is needed is a system and method for efficient caching.
5

6 SUMMARY OF THE INVENTION

7 This disclosure reveals methods and systems for responding
8 to a request for information using software processes or threads,
9 otherwise referred to herein as VGENs. In an embodiment, the
10 request for information can be a HTTP request as received through
11 a network such as the internet, however the invention herein is
12 not limited to such an embodiment, and requests can be processed
13 through other networks or non-networked environments. Similarly,
14 requests other than HTTP requests can be processed.

15 In an embodiment wherein the request includes a Uniform
16 Resource Location (URL), the URL can be used to generate a key
17 that provides an index to a hash table. The hash table
18 establishes relationships between requests or corresponding keys
19 and any software threads that have previously processed those
20 requests or corresponding keys.

21 In an embodiment, the software threads, or VGENs, are
22 capable of being modified by incorporating application code into
23 the software thread. Similarly, application code within a

1 software thread can be updated to a latest version to provide
2 uninterrupted processing of requests.

3 In an embodiment wherein a hash table is used to relate a
4 key from a URL to a software thread, if none of the software
5 threads have processed a request according to the key, a software
6 thread is used that corresponds to the non-processing software
7 thread that has been in a non-processing state for the longest
8 period of time when compared to other non-processing threads.
9 Alternately, if one or multiple software threads are identified
10 through the hash table as having processed a request according to
11 the key, the non-processing identified thread that has been in a
12 non-processing state for the longest time is selected.

13 When a VGEN or software thread is selected to process a
14 particular request for a first time, the software thread can
15 retrieve application code to process the request, byte-compile
16 the code, and incorporate the code into the software thread. If
17 persistent connections to databases, real-time data sources, file
18 systems, session manager daemons, etc., facilitate processing the
19 request, the software thread can establish and maintain such
20 persistent connections for future processing requests. The
21 software thread can also retrieve data from these persistent
22 connections, optionally process it to a more rapidly accessible
23 form and store it into the thread's local memory. This is
24 sometimes referred to as localized caching. Alternately, during

1 subsequent requests for processing, a VGEN can compare the VGEN's
2 byte-compiled version of code for a given request to an off-line
3 repository for storing application code, to determine the latest
4 version of application code. If the off-line code is a new
5 version, the VGEN can retrieve, byte-compile, and incorporate the
6 new code accordingly. The off-line repository can be any memory
7 device or device that includes a memory, is accessible to the
8 software thread, and capable of storing application code.

9 The VGENs can reside on a server or other processor that
10 receives a request, or the VGENs can reside on other processor-
11 based machines or servers that are not responsible for receiving
12 the request. A request received by one server can thus be
13 propagated to one or multiple other servers for processing by
14 VGENs.

15 Other objects and advantages of the invention will become
16 obvious hereinafter in the specification and drawings.
17

18 BRIEF DESCRIPTION OF THE DRAWINGS

19 A more complete understanding of the invention and many of
20 the attendant advantages thereto will be readily appreciated as
21 the same becomes better understood by reference to the following
22 detailed description when considered in conjunction with the
23 accompanying drawings, wherein like reference numerals refer to
24 like parts and wherein:

1 FIG. 1 is a block diagram of a system practicing the basic
2 principles of the invention for implementing VGEN engines in a
3 web server embodiment;

4 FIG. 2 shows a block diagram illustrating the updating of
5 application code in a VGEN;

6 FIG. 3 demonstrates two embodiments of VGEN engines in
7 accordance with a system according to FIG. 1;

8 FIG. 4 illustrates a system in accordance with FIGS. 1 and
9 3, wherein a request from a web server is processed by a VGEN
10 engine;

11 FIG. 5 is shows a hash table for utilization by a system
12 according to FIG. 4, wherein VGEN engines can be identified
13 through URL keys;

14 FIG. 6 provides a logic block diagram demonstrating the
15 selection of a VGEN engine for a system in accordance with FIGS.
16 1, 3, and 4.
17

18 DESCRIPTION OF ILLUSTRATED EMBODIMENTS

19 To provide an overall understanding of the invention,
20 certain illustrative embodiments will now be described; however,
21 it will be understood by one of ordinary skill in the art that
22 the systems and methods described herein can be adapted and
23 modified to provide systems and methods for other suitable

1 applications and that other additions and modifications can be
2 made to the invention without departing from the scope hereof.

3 For the purposes of the disclosure herein, a "server" can be
4 understood to include a processor, a memory (e.g. RAM), a bus to
5 couple the processor and the memory, a mass storage device (e.g.
6 a magnetic or optical disk) coupled to the processor and the
7 memory through an I/O controller, and a network interface coupled
8 to the processor and the memory. The servers may further include
9 one or more mass storage devices such as a disk farm or a
10 redundant array of independent disks ("RAID") system for
11 additional storage and data integrity. Read-only devices, such
12 as compact disk drives and digital versatile disk drives, may
13 also be connected to the servers. Servers can be understood to
14 be, for example, personal computers (PCs), SUN workstations,
15 handheld, palm, laptop, or other microprocessor controlled
16 devices for performing the operations and functions as described
17 herein and attributed to servers. Servers can be connected via
18 networks for more efficient processing of client traffic.
19 Servers in stand-alone or network configurations can operate
20 together or independently for different functions, wherein a
21 server can be designated a database server, an application
22 server, a web server, etc. As used herein, the term "server" is
23 intended to refer to any of the above-described servers that
24 further includes instructions for causing the server processor to

1 perform the functions designated and attributed to the servers
2 herein.

3 Referring now to FIG. 1, there is a block diagram 10 of a
4 system practicing the basic principles of the invention for
5 implementing VGEN engines in a web server embodiment. Although
6 the illustrated embodiments of the methods and systems can be
7 interpreted with respect to a networked environment such as the
8 internet, those with ordinary skill in the art will recognize
9 that the methods and systems are not limited to an internet
10 application and can be applied to any method or system
11 implementing caching techniques. In the exemplary internet
12 embodiment of FIG. 1, any one of well-known internet browsers
13 executing on a client 12, can execute a command to retrieve
14 requested information, including for example, a web document, web
15 page, content information, etc., wherein the information is
16 retrieved from a specified internet address that corresponds to,
17 in the illustrated embodiment, a web server 14. In an
18 embodiment, the illustrated client 12 can be a server as defined
19 previously herein. As is well-known in the art, the requested
20 information can be displayed or otherwise presented to a user of
21 the client 12 via a viewing device such as a display, screen,
22 etc., that is otherwise integrated with the client 12. In the
23 internet embodiment, user requests for information can be
24 executed via the browser on the client 12 wherein the browser

provides an interface for the user to designate a Uniform Resource Location (URL) and cause the browser to execute an Hyper-Text Transfer Protocol (HTTP) request to the web server 14, wherein in the illustrated embodiment, the illustrated web server 14 corresponds to the URL designated by the user. The web server 14 responds to the http request by transmitting the requested information to the client 12. Those with ordinary skill in the art will recognize that the retrieved information can be in the form of an HTTP object that includes plain text (ASCII) conforming to the HyperText Markup Language ("HTML"), Dynamic HyperText Markup Language ("DHTML"), Extensible Markup Language ("XML"), the Extensible Hypertext Markup Language ("XHML"), Standard Generalized Markup Language ("SGML"), etc. Additionally, the retrieved information can include hyperlinks to other Web documents, and the web server 14 can execute programs associated with the retrieved information using programming languages such as Perl, C, C++, or Java. The web server 14 can also utilize scripting languages such as ColdFusion from Allaire, Inc., or PHP, to perform "back-end" functions such as order processing, database management, and content searching. Retrieved information in the form of a web document may also include references to small client-side applications, or applets, that are transferred from the web server 14 to the client 12 with the web document and executed locally by the client 12, wherein

1 Java is one popular exemplary applet programming language. The
2 text within a web document may further include non-displayed
3 scripts that are executed by an appropriately enabled browser
4 using a scripting language such as JavaScript or Visual Basic
5 Script. Browsers can further be enhanced with a variety of
6 helper applications to interpret various media including still
7 image formats such as JPEG and GIF, document formats such as PS
8 and PDF, motion picture formats such as AVI and MPEG, and sound
9 formats such as MP3 and MIDI. These media formats, with an
10 increasing variety of proprietary media formats, can enrich a
11 user's interactive and audio-visual experience as a web document
12 is presented through the browser at the client 12.

13 Although the FIG. 1 system illustrates a client 12 and
14 server 14 configuration, those with ordinary skill in the art
15 will recognize that the methods and systems herein can be applied
16 to a configuration wherein the illustrated client 12 can also be
17 a server such as the web server 14 illustrated in FIG. 1. For
18 example, application logic executed by a first server (e.g.,
19 depicted as the client 12) having the same attributes and
20 functionality of the illustrated web server 14, can issue a HTTP
21 request to a second web server (e.g., depicted as the web server
22 14), wherein the application logic can be executed on the second
23 server to produce, for example, XML results. In this example
24 embodiment, the XML results from the second server can be

1 transferred to the first server and thereafter to the initial
2 requesting entity. In other embodiments, multiple numbers of
3 servers such as the web server 14 of FIG. 1, can make requests of
4 each other, wherein the subsequent server's results can be
5 transferred to a requesting server. In different embodiments,
6 the requesting and executing servers can be configured the same
7 or differently while remaining in the scope of the invention.

8 Referring again to the methods and systems of FIG. 1, the
9 illustrated web server 14 processes the request from the client
10 12 using at least one VGEN Engine 16, otherwise referred to as a
11 VGEN, wherein in the FIG. 1 embodiment, there are N VGENs 18,
12 where N is a positive integer. The illustrated VGENs 18 are
13 software processes or threads that can be executed separately
14 from the web server functionality as such web server
15 functionality is previously described herein, although a VGEN can
16 reside on the web server 14 and be processed by the web server
17 processor. Alternately, a VGEN 16 can reside on a server that is
18 distinct from the web server 14 but can be in communication with
19 the web server 14.

20 In the illustrated embodiments, VGEN processes 18 execute
21 continuously and can execute scripts, run applications, connect
22 to databases, etc. In an embodiment, a standard VGEN 16 at
23 system initialization includes applications and basic
24 instructions for executing tasks such as loading data

(applications, database data, XML pages, CGI scripts, etc.) into the VGEN 16, parsing data, processing applications, byte-compiling data, establishing and maintaining persistent connections, etc., although such base functionality is merely provided for illustration and not limitation, and those with ordinary skill in the art will recognize that other embodiments of VGENs 18 can include additional functionality or less functionality while remaining in the scope of the invention.

The illustrated VGENs 18 can retrieve, cache, and distribute dynamic and other content, and generate requested web document ("page") information and forms. Those with ordinary skill in the art will recognize that the illustrated VGENs 18 can be implemented in hardware or software, and in either embodiment, VGENs can access cache memory, wherein such cache may be resident on the web server 14 or on another server, computer, etc. In one embodiment, every VGEN 18 maintains a connection to a dedicated cache area to increase system robustness, however in alternate embodiments, VGENs 18 can share cache areas or segments. For example, a single cache area can be partitioned or otherwise divided for the number of VGENs. In an embodiment wherein VGENs process similar or identical information, multiple VGENs can access a common cache area. In yet another embodiment, a common cache area can be replaced with the ability of each VGEN to access the cache of other VGENs.

1 As indicated by FIG. 1, the illustrated VGENs 18 can also
2 access Application Logic 20 that is typically located on memory
3 that can reside on the web server 14, another networked machine,
4 or otherwise accessible memory; and, the application logic 20 can
5 include, for example, CGI Pages, XML Pages, Server Pages, etc.
6 As mentioned previously, a basic VGEN 16 includes logic for byte-
7 compiling applications, and therefore when an illustrated VGEN 16
8 retrieves application logic 20 from memory, the illustrated VGEN
9 16 can byte-compile and cache the application logic 20 into the
10 VGEN 16 logic for subsequent, rapid execution of the application
11 in responding to subsequent requests for the application 20.

12 As indicated in illustrated system of FIG. 1, the
13 application logic 20 can be facilitated through a persistent
14 connection with a database 22, real-time data 24, a session
15 manager daemon 26, a file system 28, etc., although those with
16 ordinary skill in the art will recognize that persistent
17 connections can be maintained with other devices or objects
18 without departing from the scope of the invention. The
19 illustrated real-time data 22 can be provided through an
20 input/output port or other source, while the illustrated file
21 system 28 can represent the web server's file system, or another
22 file system.

23 When an illustrated VGEN 16 byte-compiles application code
24 20 and incorporates the code into the VGEN 16, the VGEN 16 can

1 maintain the persistent connections as deemed necessary by the
2 application 20. As requests are obtained by the web server 14
3 and distributed to VGENs 18, the respective VGENs 18 can
4 aggregate byte-compiled application code 20 and associated
5 persistent connections. By distributing the requests across the
6 various VGENs 18, respective VGENs 18 can be requested to execute
7 subsequent requests for respective application code, thereby
8 providing an increased cache system without the disadvantages of
9 continually compiling and caching application logic 20 and
10 maintaining persistent connections to every data source from
11 every VGEN.

12 The FIG. 1 system also provides a mechanism to provide
13 updates to application code 20 without disrupting the web server
14 14 in its ability to handle requests from the illustrated client
15 12. As indicated previously, the illustrated VGENs 18 include
16 basic functionality to retrieve application code 20; however,
17 this logic also includes the ability to verify application code
18 that is presently byte-compiled within the selected VGEN 18.
19 FIG. 2 shows a block diagram indicating a logic flow for the
20 illustrated VGENs 18 of FIG. 1 that allows for the uninterrupted
21 update of application code 20.

22 As indicated in FIG. 2, when a request is received by the
23 web server 14 of FIG. 1, an illustrated VGEN 18 of FIG. 1 is
24 selected to process the request 100 as shall be described herein.

1 The selected VGEN can verify whether the application code 20 to
2 execute the request resides within the selected VGEN 102, and if
3 the selected VGEN does not include the application code 20, the
4 application code 20 is retrieved from memory, disk, etc. 104,
5 byte-compiled, incorporated into the VGEN, and utilized to
6 execute the request 106. Returning to 102, alternately, if the
7 selected VGEN determines that the selected VGEN does include the
8 application code 20 to process the request, the selected VGEN
9 retrieves information about the latest version of the application
10 code 20 from memory, disk, etc., 108 and compares the newly
11 retrieved application code information with the application code
12 information from the selected VGEN 110. If the newly retrieved
13 application code information indicates that the memory, disk,
14 etc., includes an updated or more recent version of the
15 application code as compared to the application code in the VGEN
16 112, the newer application code 20 can be retrieved from memory,
17 disk, etc., 104 byte-compiled, incorporated into the selected
18 VGEN to replace the previous version of the application code, and
19 utilized to execute the request 106. Returning to 112, if the
20 application code in memory, on disk, etc., is not newer than the
21 application code that is part of the selected VGEN, the
22 application code is not retrieved from memory, disk, etc., the
23 selected VGEN does not change, and the existing selected VGEN is
24 utilized to process the request 106.

1 One with ordinary skill in the art will recognize that
2 certain requests from clients occur with greater frequency than
3 other requests; therefore, for the illustrated systems and
4 methods, multiple VGENs 18 may include the same or similar
5 application logic 20 and associated persistent connections to
6 facilitate multiple and/or simultaneous requests for the same or
7 similar information. Additionally, although the illustrated
8 systems and methods retrieve and replace newer versions of
9 application code 20 to update the respective VGEN 18, in other
10 embodiments, multiple versions of application code can be
11 incorporated into a VGEN 18.

12 Referring to FIG. 3, there is a diagram illustrating two
13 embodiments of VGEN implementations in accordance with a system
14 according to FIG. 1. As illustrated in FIG. 3, the web server 14
15 can include a VGEN object 30 that thereafter includes a VGEN data
16 structure 32 that maintains statistics for a VGEN 16. In an
17 embodiment, the VGEN object 30 can be a thread-safe C++ module
18 that interfaces with the web server's 14 high performance
19 extension interface. In the illustrated system, there is a VGEN
20 data structure 32 for every VGEN 16, but those with ordinary
21 skill in the art will recognize that other implementations and/or
22 structures can accomplish the same result of maintaining data
23 related to a VGEN 16, without departing from the scope of the
24 invention. In the illustrated system, the VGEN data structure 32

1 can include information based on the current processing status of
2 a VGEN 16, the length of time a VGEN 16 has been processing data,
3 security permissions for a VGEN 16, application code or data in a
4 VGEN, persistent connection information, etc. In one embodiment
5 illustrated in FIG. 3, the VGEN data structure 32 maintains data
6 on a VGEN 16a that resides within the web server 14.

7 In another embodiment illustrated in FIG. 3, the VGEN data
8 structure 32 can maintain information on a VGEN 16b that resides
9 on a server 34 that is distinct from the web server 14. The
10 invention herein is not limited to either of the two embodiments
11 indicated in FIG. 3, and can include combinations of the two
12 illustrated embodiments wherein some VGENs 16a reside on the web
13 server 14, while other VGENs 16b reside on any of various other
14 servers 34. Similarly, in the illustrated embodiment, the VGEN
15 object 30 and VGEN data structure 32 can reside at locations
16 other than the web server 14. One with ordinary skill in the art
17 will therefore recognize in referring to FIGs. 1 and 3, that the
18 system of FIG. 1 that displays the web server 14, VGENs 18,
19 application code 20, databases 22, etc., as distinct elements,
20 such display is merely for illustration purposes, and the
21 elements therein can be combined and incorporated as part of the
22 web server 14, or can be distributed across multiple servers or
23 other platforms, without departing from the scope of the
24 invention.

Referring now to FIG. 4, there is shown a basic methodology 40 indicating the processing of a request received by the web server 14 of FIGs. 1 and 3. As FIG. 4 indicates, in the illustrated systems, the request is transferred to a hash table 42 and VGEN engine list 46 that utilize information from the request. Referring now to FIG. 5, an exemplary hash table 42 is presented to indicate the methodology by which the web server 14 of FIGs. 1 and 3 determines which of the various VGENs should process a request.

As indicated previously with respect to FIG. 1, the web server 14 can receive a HTTP request from the client 12 that includes a URL. The illustrated systems and methods can utilize the URL information in the HTTP request to generate a key 44 for the hash table 42 illustrated in FIG. 4. For example, a URL parsed from the HTTP request can have the general format of: http://host:port/path;params?query. In an embodiment, a hash table key 44 can be generated using, for example, the "path" and "params" values only, although such example is provided merely for illustration and not limitation. Once generated, the key 44 can be related to one or more VGENs 18 associated with the key 44. Although FIG. 4 illustrates the relationship between the key 44 and VGENs 18 in tabular or matrix format, those with ordinary skill in the art will recognize that such relationships between keys 44 and VGENs 18 can be formulated using, for example,

1 queues, linked lists, or any other well-known data structure for
2 formulating relationships or otherwise associating data elements,
3 wherein such mechanisms can be utilized in cooperation with, or
4 to the exclusion of, a hashing algorithm.

5 In the illustrated embodiment, once the methods and systems
6 determine the VGEN(s) 18 associated with the key 44 computed from
7 the web server's request, the associated VGENs 18 are polled to
8 determine an associated VGEN 18 available to process the request.

9 Referring back to FIG. 4, VGEN availability can be determined
10 using information from the VGEN data structures 32 related to the
11 associated VGENs 18. Similarly, the VGEN data structure 32
12 information can be provided in part from a VGEN Engine List 46
13 that monitors the activities of all VGENs 18 and maintains a list
14 of available (i.e., currently non-processing) VGENs 18. In an
15 embodiment, the Engine List 46 is a doubly linked-list of
16 available VGENs, although such embodiment is provided for
17 illustration and not limitation.

18 In the illustrated system of FIG. 5, the VGENs associated
19 with the URL key and identified by the hash table 42, are polled
20 to determine an available VGEN 18. In the illustrated systems,
21 the first available, associated VGEN 18 is selected to process
22 the request, and the processing result is thereafter provided to
23 the web server 14 for response to the client 12 as indicated in
24 FIG. 1.

Referring now to FIG. 6, there is a block diagram 50 of the methodology to select VGENs 18 in the illustrated systems, for processing a request and distributing application code and cached information amongst the various available VGENs 18 to increase processing efficiency. As indicated in FIG. 6, for the illustrated systems and methods, a request is received from the web server 52, and from the request URL information, a hash table key is created 54. If the key does not exist in the hash table 56, a VGEN is selected from the top of the VGEN engine list 58. The hash table is thereafter modified to include the new key and selected VGEN 60, whereupon the selected VGEN is removed from the engine list 62, and the request is assigned to the selected VGEN for processing 64. Once the selected VGEN processes the request, the selected VGEN is returned to the bottom of the engine list 66 and the processing results are transferred to the web server 68.

By assigning VGENs to new keys, extracting from the top of the VGEN engine list, and returning the assigned VGENs to the bottom of the VGEN engine list, load balancing is achieved between the various VGENs.

Referring again to FIG. 6, if the illustrated methods and systems determine that the key is located in the hash table 56, the list of VGENs (or single VGEN) associated with the key are determined 70, wherein the associated VGENs are polled to determine the an available, associated VGEN 72. In one

embodiment, any such available associated VGENs can be ordered by time of last use or idle time, or some other designated criteria.

If all associated VGENs are unavailable, a new VGEN is associated with the key by extracting an available VGEN from the top of the engine list 58. The illustrated system processing thereafter continues by updating the hash table 60, removing the selected VGEN from the engine list 62, assigning the request to the selected VGEN 64, waiting for completion of processing, and upon completion of processing, placing the selected VGEN at the bottom of the engine list 66 and returning the processing results to the web server 68.

Alternately, if one of the associated VGENs is available as determined by either the engine list and/or the VGEN data structures, the first such available VGEN is determined to be the "selected" VGEN and is accordingly removed from the engine list 62, the request is assigned to the selected VGEN 64, and upon completion of processing, the selected VGEN is returned to the bottom of the engine list 66 and the processing results are returned to the web server 68. Those with ordinary skill in the art will recognize that several of the processes as illustrated in FIG. 6, can be combined or otherwise interchanged without departing from the scope of the invention. For example, the processing results can be returned to the web server 68 while, before, or subsequent to, the VGEN being placed at the bottom of

1 the engine list 66. Those with ordinary skill in the art will
2 also recognize that the diagram of FIG. 6 did not include updates
3 between the engine list and VGEN data structure such relationship
4 was previously defined in FIGs. 3 and 4, and thus the
5 availability of VGENs as shown in FIG. 6 can be derived from a
6 VGEN data structure of FIGs. 3 and 4 via the engine list that
7 communicates with the VGEN data structures, or in alternate
8 embodiments, VGEN availability can be obtained directly from the
9 engine list. Additionally, the logic flow of FIG. 6 is designed
10 to accompany an embodiment of the invention as indicated in FIG.
11 1, and therefore the illustrative logic of FIG. 6 can be altered
12 for alternate embodiments of the invention.

13 As the methods and systems of FIGs. 1-6 illustrate, the
14 number of VGENs can be established (i.e., expanded or reduced) to
15 facilitate the processing of requests according to system
16 requirements. For example, if requests are being received
17 wherein the Engine List 46 of FIG. 4 cannot accommodate the
18 request, more VGENs can be added to ensure a list of available
19 VGENs for all requests. As indicated previously, in many
20 embodiments, there can be a likelihood that several VGENs can
21 have the same information to process more popular requests. In
22 some embodiments, each VGEN has a dedicated cache or section
23 thereof, while in other embodiments, VGENs that can process
24 similar requests can share cached information.

1 One of several advantages of the present invention over the
2 prior art is that distributed caching of information can be
3 accomplished for efficient processing of information to prevent
4 an overloading of system resources that would otherwise occur
5 without such distribution.

6 What has thus been described are methods and systems to
7 distribute cached information for efficient processing without
8 overloading system resources. In an embodiment, VGENs can be
9 implemented as software engines that can reside on a web server
10 or another server that responds to requests for information in a
11 networked environment. A hash table can identify whether a VGEN
12 exists with cached information to process the request. If an
13 available VGEN engine includes the cached information, the
14 available VGEN is assigned to process the request; otherwise,
15 another available VGEN can be selected to process the request,
16 wherein the VGEN is augmented to include the cached information
17 and the hash table is updated to reflect the VGEN statistics.
18 VGENs can maintain persistent connections to databases, file
19 systems, session manager daemons, real-time data sources, etc.,
20 to facilitate the processing of requests.

21 Although the present invention has been described relative
22 to a specific embodiment thereof, it is not so limited.
23 Obviously many modifications and variations of the present
24 invention may become apparent in light of the above teachings.

1 For example, many of the elements of the illustrated systems and
2 methods can be combined without departing from the scope of the
3 invention. Although the illustrated systems utilized hash
4 tables, other methods of relating a request to a VGEN can be
5 implemented. The illustrated VGENs were implemented in software
6 to access cache memory areas, however the VGENs can be
7 implemented in hardware. Although the systems and methods
8 utilized persistent connections between the VGENs and databases,
9 file systems, etc., some embodiments may not require or utilize
10 persistent connections. In some embodiments, application code
11 may not be updated as was indicated in the illustrated systems.
12 Although the illustrated systems and methods distributed the
13 requests among available VGENs utilizing a last-used criterion,
14 other methods or criteria for distributing the request or
15 otherwise performing load balancing, can be practiced. For
16 example, total processing time can be utilized to select an
17 available VGEN.

18 Many additional changes in the details, materials, steps and
19 arrangement of parts, herein described and illustrated to explain
20 the nature of the invention, may be made by those skilled in the
21 art within the principle and scope of the invention.
22 Accordingly, it will be understood that the invention is not to
23 be limited to the embodiments disclosed herein, may be practiced
24 otherwise than specifically described, and is to be understood

1 from the following claims, that are to be interpreted as broadly
2 as allowed under the law.